Intel Mash Maker: Join the Web

Rob Ennals
Intel Research
2150 Shattuck Avenue
Penthouse Suite
Berkeley, CA 94704, USA
robert.ennals@intel.com

Eric Brewer
Intel Research
2150 Shattuck Avenue
Penthouse Suite
Berkeley, CA 94704, USA
eric.a.brewer@intel.com

Minos Garofalakis*
Yahoo Research
2821 Mission College Blvd
Santa Clara, CA 94, USA
minos@yahoo-inc.com

Michael Shadle
Software Solutions Group
Intel Corporation
5200 NE Elam Young Parkway
Hilsboro, OR 97124, USA
michael.shadle@intel.com

Prashant Gandhi
Intel Research
2200 Mission College Blvd
Santa Clara, CA 95054, USA
prashant.gandhi@intel.com

ABSTRACT

Intel®Mash Maker is an interactive tool that tracks what the user is doing and tries to infer what information and visualizations they might find useful for their current task. Mash Maker uses structured data from existing web sites to create new "mashed up" interfaces combining information from many sources.

The Intel®Mash Maker client is currently implemented as an extension to the FireFox web browser. Mash Maker adds a toolbar to the browser that shows buttons representing enhancements that Mash Maker believes the user might want to apply to the current page. An enhancement might combine the data on the page with data from another source, or visualize data in a new way. Mash Maker is intended to be an integral part of the way the user browses information, rather than being a special tool that a user uses when they want to create mashups.

In order to create mashups from normal websites, Mash Maker must first extract structured data from them. If the web site does not provide RDF data, then Mash Maker extracts structured data from the raw HTML using a community-maintained database of *extractors*, where each extractor describes how to extract structured data from a particular kind of web site.

Categories and Subject Descriptors

H.4.3 [**Information Systems**]: Information Browsers

General Terms

Management, Design, Human Factors

Keywords

Mashup, Data integration, Personalization, Visualization

*Work done primarily while at Intel Research

1. INTRODUCTION

Historically, the process of writing new queries and creating new graphic interfaces has been something that has been left to the experts. A small set of experts would create applications, and all users would have to make do with what was available, even if it did not quite fit their needs.

Mashups are an attempt to move control over data closer to the user and closer to the point of use. Although mashups are technically similar to the data integration techniques that preceded them, they are philosophically quite different. While data integration has historically been about allowing the expert owners of data to connect their data together in well-planned, well-structured ways, mashups are about allowing arbitrary parties to create applications by repurposing a number of existing data sources, without the creators of that data having to be involved. The importance of mashups is arguably more political and cultural than technical. Mashups are about the "democratization of innovation" [11].

Intel®Mash Maker is a project within Intel Research that is aiming to push the mashup envelope a few steps further. Previous work in mashups has followed a model in which a reasonably skilled user uses a special interface to visually compose information from different data sources, creating a new mashed-up application that can then be used by other users. Although this approach has empowered a new class of semi-skilled users to create a vast number of customized applications, specially tailored for particular tasks, or particular groups of people, we believe that the concepts can be taken much further. With Intel®Mash Maker, our intention is that normal users should be able to easily create applications and interfaces that are specially customized not only for them, but for the exact task they are performing at that moment. Our aim is mashups for the moment, on demand.

Intel®Mash Maker does this by making mashup creation part of the normal browsing process. Rather than having a reasonably skilled user create a mashup in advance as a mashup site that other users browse to, Mash Maker instead creates personalized mashups for the user inside their web browser. Rather than requiring that a user tell a mashup tool what they want to create, Mash Maker instead watches what information the user looks at, correlates the user's behavior with that of other users, and guesses a mashed up application that the user would find useful, without the user even having to realize they wanted a mashup.

Mash Maker is currently implemented as an extension to the

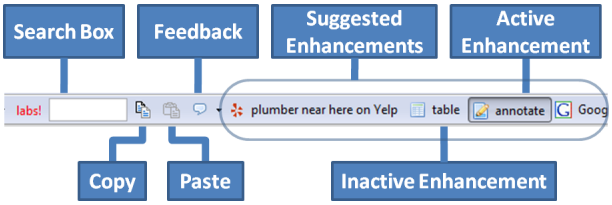

Figure 1: The mashup bar

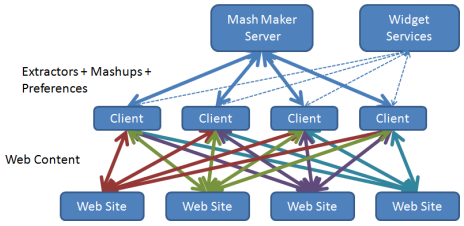

Figure 2: The Mash Maker client and server

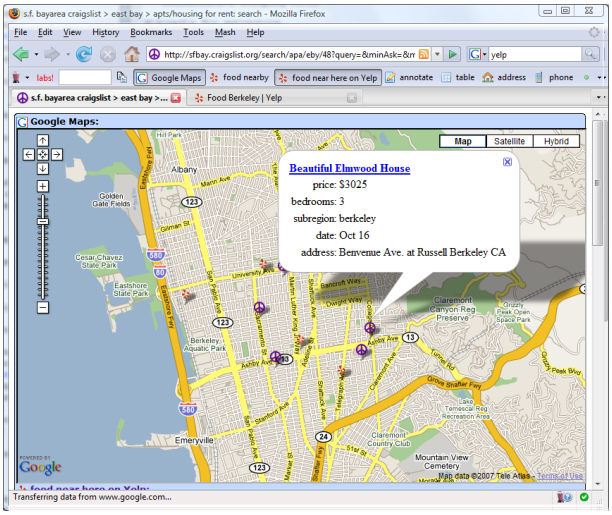

Figure 3: Visualizing data on a map

FireFox web browser, with versions planned for other browsers. As the user browses the web, the Mash Maker toolbar displays buttons representing enhancements that Mash Maker thinks the user might want to apply to their current page (Figure 1). The user need simply turn on some combination of these enhancements to create a new mashup; e.g., to plot all items on a map, the user can click on the Google Maps button.

In addition to suggesting known enhancements defined by other users, Mash Maker will also suggest new enhancements that it has created by filling in *gaps* in known enhancements (Section 4.3). Similarly, a user can create a composite mashup by turning on several generic enhancements (e.g. good restaurants + crime level + map).

If a user knows what enhancement they want and Mash Maker does not suggest it then the user can use a simple copy and paste interface to show Mash Maker a pair of web sites that the user would like Mash Maker to combine (Section 4.2).

If a web site source exports its data in a structured form such as RDF then Mash Maker can use this, otherwise Mash Maker must extract structured data from the raw HTML. Mash Maker consists of two key parts: the client, which is a browser extension that allows a user to create mashups as part of their normal browsing process; and the server, which stores *extractors* that tell Mash Maker how to extract structured data from normal web sites (Figure 2). The server operates like a wiki, allowing any user to edit the extractor for a page.

We believe that Mash Maker offers a radically new approach to querying and visualizing data:

- **Mashups for me, right now, on demand.** Mash Maker allows an unskilled used to create mashups that are tailored not only for them, but tailored for the task they are performing right now.

- **The Mashups come to you.** Mash Maker watches what the user does and tries to suggest mashups that the user will like.

- **Mashing is browsing.** Mash Maker does not require a user to use a special mashup creation interface, or browse to special mashup sites. Instead, Mash Maker augments the familiar web browsing interface that the user already uses to browse data, and enhances this with mashed up information.

- **Rely on the community to structure your unstructured data.** Mash Maker can mash up data from web sites that have no structured data API. It does this by maintaining a centralized community-maintained database of *extractors*. Any user can edit this database to create an *extractor*, describing how to extract structured data from a particular kind of web page.

This paper should be approached as an overview paper. Many of the topics discussed in this paper contain considerable subtleties that we do not have space to explore fully.

2. A QUICK TOUR OF MASH MAKER

We will begin with a couple of example usage scenarios for Intel®Mash Maker, illustrating some of the concepts that we discuss in this paper:

2.1 Showing things on maps

Over half of the mashups listed on ProgrammableWeb.com involve plotting things on a map. Since this is a common mashup scenario, we will use that as our first example, by showing how a user, Alice, creates the classic "Craigslist houses on a map" mashup using Mash Maker.

Alice browses to the Craigslist apartment listing, as she would normally, and browses the apartments that are available. Mash Maker notices that the page contains items with addresses, and so displays a "Google Maps" button that Alice can use to visualize this data on a map. Alice is interested in an apartment that has good restaurants nearby, so she opens another window and searches for restaurants on Yelp. Mash Maker notices that Alice is interested in restaurants and so updates the mashup bar for the Craigslist page to suggest adding a list of restaurants in the area.

Alice clicks on this new suggestion and Mash Maker responds by inserting information about Yelp restaurants directly into the Craigslist page. Mash Maker found the Yelp restaurants by passing the current location and the topic "food" as arguments to a form on the Yelp website. In effect, Mash Maker has performed a join on the data in Craigslist and Yelp, while obtaining the data through the standard web interface.

Alice also clicks on the "Google Maps" button to see all this data visualized on one map (Figure 3). Finally, Alice turns on the "annotate" enhancement. This is a built-in enhancement that can be suggested for any page that contains items with their own URLs.

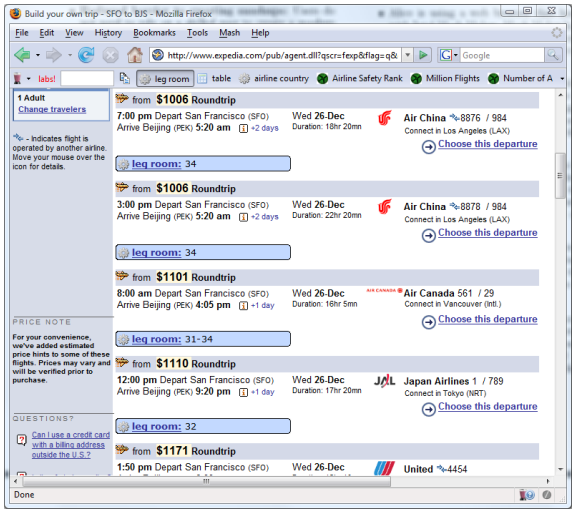

Figure 4: Flights annotated with legroom

The annotate enhancement augments each item on the page (in this case apartments) with an interactive widget that allows Alice to attach a persistent personal note. Items are identified by the (normalized) URLs, and so the same note will appear for each item if it appears on a different page, provided the item's URL is the same. As demonstrated by the "annotate" enhancement, not all enhancements bring in data from other web pages. Some enhancements add new UI features or services that the user might want.

2.2 Information about Flights

As a second example, we will demonstrate the general principle of "copy and paste mashups" by having a user, Bob, add leg room information to Expedia flights.

Bob browses to Expedia and searches for flights. Bob is concerned that he should book a flight that will give him a lot of leg room, but Mash Maker is not currently suggesting legroom. Bob is disappointed that Mash Maker has not given him a button to add leg room to the page, so he browses to a page giving the typical leg room for different airlines. Mash Maker has still not guessed what Bob wants, so he clicks "copy" on the leg room and "paste" on the flight listing, to tell Mash Maker that he would like to augment the page about flights with information from the page about legroom. Mash Maker brings up a dialog box, asking Bob how the data should be connected. Mash Maker has correctly guessed that the data should be equi-joined based on the "Airline" column, but is not sure with which column to annotate the flights. Bob selects "economy legroom", gives the new enhancement a brief name and description, and clicks "publish" to tell Mash Maker that this enhancement should be suggested to other users. Mash Maker turns the new enhancement on automatically, giving the display in Figure 4.

3. THE BASIC APPROACH

The examples in Section 2 demonstrated several of the features of the Intel® Mash Maker user interface. The user can turn on and off any combination of enhancements by clicking on buttons on the mashup toolbar (Figure 1). The user can also search for a specific enhancement by typing keywords into the search box. If Mash Maker does not suggest appropriate enhancements then the user can show Mash Maker what they want by viewing web pages about the topic they are interested in, or by using copy and paste to explicitly tell Mash Maker what they want to combine.

Mash Maker allows the user to pick and choose the information and visualizations that they would like to combine together. Going back to the example from Section 2.1, there are many different apartment listing sites, many different local search services, and many different mapping services. With Mash Maker, a user can pick any combination of these services by simply browsing to their preferred apartment listing service, and then turning on the enhancements for their favorite local search service and mapping service. It is not necessary for any pair of these services to have been combined previously, so long as other users have previously shown Mash Maker the generic ways that these sites can be combined with other sites (Section 4.2).

When enabled, an enhancement adds additional content to the current web page (Figure 4). This information is visualized using a widget, which may be static or interactive. Enhancements are non-side-effecting. Clicking an enhancement button can only add information to the page — it cannot perform externally visible actions. If an enhancement wants to perform actions, then it must do this by inserting a widget that the user can use to perform such actions. For example, rather than having an "add to my calendar" enhancement (as in Operator [18]), one would instead have a "calender operations" enhancement that adds an "add to my calendar" button to every event on the page.

Internally, Mash Maker describes its enhancements using an underlying functional programming language [5].

Previous mashup tools [8, 19, 20, 21] have taken a server-based approach in which a mashup server retrieves data from other sites and uses this to create a new web site that hosts the mashup. Mash Maker instead runs almost entirely on the client, running as an extension to the user's web browser. This client-based approach has advantages for data access, privacy, performance, and user experience. Since Mash Maker is a browser extension, Mash Maker can see everything the browser can see, including local files, information on the intranet, information requiring a login, and active content generated by Javascript. There are also privacy and performance advantages, since Mash Maker does not need to ship data to and from a central server in order to create the user interface. Finally, running as an integrated part of the browser allows Mash Maker to present a more pervasive user experience, in which mashups are an integral part of the way the user looks at information.

A key driving principle of Mash Maker is that mashups should be personal. Rather than using fixed mashups created by other users, an unskilled user should instead be provided with mashups that have been created specially for them, and for the task that they are currently performing. Mash Maker looks at the information that the user browses, and the mashups that the user has turned on in the past, and uses this to guess what mashups this particular user might want right now, which it then suggests on the mashup bar. Since Mash Maker runs on the client, it can mine sensitive private information without leaking it to third parties. In future versions, we plan to make use of physical sensor information such as location, time, device type, and inferred activity, to improve the suggestions that Mash Maker makes.

3.1 The Server

In order to create such mashups, Mash Maker needs to understand the meaning of web pages. In an ideal world, all web sites would expose the structured databases that underly their sites, making it easy to integrate the data, however this ideal world has not yet arrived. It is thus necessary, at least for now, that we use more ad-hoc techniques to understand the meaning of web pages.

Mash Maker understands the meaning of web pages using a col-

laboratively maintained database of *extractors*. An extractor describes how to extract structured information from the raw HTML of a particular kind of page. For example, the extractor for Craigslist apartments says how to find an apartment on the page, and how to extract each of the properties of an apartment.

The Mash Maker server is influenced by wikis such as Wikipedia. Like a wiki, the Mash Maker server allows any user to edit the extractors for any web site. To avoid vandalism, Mash Maker allows high profile or sensitive pages to be locked down so that they can only be edited by trusted users. Mash Maker also provides a complete version history, allowing users to roll back previous edits if vandalism has occurred. Section 3.4 briefly discusses some of the security issues relating to bad data.

In addition to being able to tell us how to extract meaning from a page, the Mash Maker server also stores information about how the page is parameterised. For example, if we have a page about "England", then the server can tell us that the page is parameterized by a country, and that the argument is encoded as a form parameter of the URL. A related URI-comprehension mechanism is also used for normalizing URIs that are textually different but refer to the same resource. This information is provided by a collection of *arg handlers*, which are managed similarly to extractors.

The Mash Maker browser extension includes an extractor editor, allowing any user to edit the semantic extractor for the page they are currently browsing by opening the extractor editor side bar. Indeed Mash Maker will prompt the user to do this if it does not understand the meaning of the current page.

3.2 Suggestions

Mash Maker chooses which enhancements to suggest using an ad-hoc algorithm that assigns weights to enhancements based on a number of factors. The main factor affecting the weight of an enhancement is how recently and how often the current user and other users have applied that enhancement to pages similar to the current page. For each extractor/enhancement pair, the Mash Maker server maintains a record of how often and how recently the user and the community as a whole have applied that enhancement to pages described by that extractor.

The suggestion algorithm also uses a number of ad-hoc heuristic rules to improve suggestion weights, including favoring information from sites that the user has viewed recently, and taking account of explicit votes for and against particular enhancements by users. The current suggestion algorithm is quite crude and we believe there is much potential for improvement. We plan to improve it in future work.

3.3 Copyright

We have no interest in using content in ways that the creators disapprove of. However, it is impractical to ask permission from every site in advance, since we don't know what content our users might wish to combine. Our approach is to assume that a small amount of data extraction from a website is probably harmless. If we see that a data source is being used a lot, then we will contact the owners of that content to ask them if and how they would like their content to be used, and store this information on the central server. For content for which we do not yet have an agreement, we throttle the rate at which Mash Maker extracts data until we know what the content owner wants. Our hope is that, just as most web sites like being listed by Google, most content owners will appreciate the additional exposure Mash Maker provides for their content.

3.4 Privacy and Security

Giving unskilled users the power to combine data sources in previously unexpected ways opens up a number of issues for privacy and security. If one applies an enhancement to a page that contains private information, then that enhancement could cause that information to become visible to a third party. For example the "Google Maps" enhancement sends all addresses on the page to Google, which might not be acceptable if they were the addresses of confidential locations. There are several ways we try to address this problem. First, all enhancements are manually checked by trusted "moderator" users, before they can be suggested to other users, to make sure that they are not obviously malicious. Second, the Mash Maker server has facilities for marking data on a page that should be considered private, and should not be passed outside the client. Third, some standard classes of confidential data, such as passwords, and credit card numbers, can be easily detected and blocked from being passed outside the client. This is an area of active research, and we do not yet have a perfect solution.

A similar issue is the ability of Mash Maker to track what users do. One of the goals of Mash Maker is that the client should tell the server as little as possible about what the user is doing. In particular, when the client requests extractors from the server, it does so for an entire domain rather than a particular page, and does not send identifying information (while we could log IP addresses, we intentionally avoid doing so). Moreover, since the Mash Maker provides extractors using an anonymous, cacheable, REST API, a group of users could potentially hide their behavior from Mash Maker by accessing it through a shared proxy. Since a user may have multiple devices and browsers that they wish to be synchronized, Mash Maker will, by default, store on the server a record of what mashups the user has applied to particular general kinds of page. The user can turn this off if they prefer.

3.5 Query Optimization

Extracting data from web pages is often a very inefficient way to access a data source. Our hope is that the popularity of mashup tools will encourage an increasing number of content providers to provide high level access to their data, through interfaces such as SPARQL [23]. If we know that the data provided by a site is not private, and that it is okay for us to cache it, then the Mash Maker server will cache it in a high-level database on the Mash Maker server. The Mash Maker client can use this database to obtain data using efficient high-level queries. For example, to retrieve the head of state of 100 countries using Wikipedia pages would require downloading 100 pages if accessed directly, or doing a small database query to the cache on the Mash Maker server. We believe that further research is needed on the optimization of cross-provider queries of web content.

4. MAKING MASHUPS

The enhancements suggested by Mash Maker take several forms, from simple linked data, through to mashups inferred from user behavior, and new visualizations. Our experience so far is that, while most users limit themselves to turning on combinations of previously defined enhancements, one only needs a fairly small number of more skilled users to create enhancements for all users to benefit, since all the enhancements these users create can be used by other users.

4.1 Linked Data

The simplest kind of enhancement is one that just follows a link on the page and inserts information that it finds there. If an item on a page contains a URL for another page, then Mash Maker will automatically provide enhancement buttons for annotating the current page with information described on the linked page.

4.2 Copy and Paste

If the user knows what mashup they want, and Mash Maker does not automatically suggest it, then they can teach Mash Maker new connections between web sites using a simple "copy" and "paste" interface. The user clicks "copy" on the *source* page that they would like to use information from, and then clicks "paste" on the *host* page that they would like to add this information to. For example, in Section 2.2 the user copied information about airlines and pasted it into a page listing flights with those airlines. Mash Maker will try to guess how the data should be combined, based on the property and form argument types for the two pages, and the behavior of past users. The most common ways to combine pages are to do a simple join of the data, and/or pass data from the host page as a form argument to the source page. If Mash Maker guesses wrong then the user can manually specify how to combine the data.

Of course, if the user has to edit the enhancement manually then we are essentially back to the same difficulty level as specifying a query in a database. Mash Maker thus tries where possible to avoid the user having to do this, either by guessing how to combine data, or by suggesting enhancements created by previous users.

4.3 Filling in the Gaps

When creating an enhancement using copy and paste, one can leave *gaps* in the definition that can be filled in later. These gaps correspond to function parameters in the underlying functional language [5]. Mash Maker can fill in gaps with information from pages the user looked at recently. For example, in Section 2.1 we used a local search enhancement that allowed Mash Maker to guess what the user was interested in. In that case, Mash Maker guessed that the user wished to search for "food" because Mash Maker had seen this was the "search term" property of a previous page. It is also possible for a user to fill in the gaps explicitly, by clicking on the button for the enhancement template, and entering the arguments directly.

4.4 Applicability of an Enhancement

Once a user has defined a new enhancement, Mash Maker can suggest the enhancement for any pages that are *similar* to the original host page. Early versions of Mash Maker could potentially suggest an enhancement for any page that had the properties that were required by the enhancement, or that had applicable enhancement that could produce such properties. However we found that this caused Mash Maker to suggest a lot of inappropriate enhancements. More recent versions allow a user to restrict the classes of items for which a particular enhancement should be suggested. For example, one might constrain the "legroom" enhancement to only be applicable to "flights".

4.5 Visualization Widgets

Mash Maker visualizes data added to a page using *visualization widgets*. Widgets range from simple static widgets such as static text and images, through to right interactive widgets such as maps. A user can write a new widget by creating a normal web page that exposes a Javascript function called `mashmaker_widget`. Mash Maker renders a widget by inserting this page as an iframe [12] and passing the javascript function a value representing the RDF data that the widget should visualize. Widgets can communicate with other web services. For example the *annotate* widget communicates with a web service that stores personal notes written by users, and the *google maps* widget communicates with Google Maps.

4.6 Schema Matching

Since content providers are not always consistent in the names they use to refer to the same thing, Mash Maker allows users to interactively teach Mash Maker which strings and URLs should be considered equivalent. If a mashup is trying to join together two data sets and can't find a connection then it will insert a button saying "click to connect this". If the user clicks this button then they are presented with an interface that allows them to say what the key should have been matched to. As with all other metadata, such equivalences are edited collaboratively, and are shared with all users.

More generally, combining information from arbitrary web pages, decoded using extractors written by different authors, is a potentially very difficult schema-matching problem. We have so far only touched on the issues that need to be addressed. While our current simple implementation is already able to do a good job in many cases, we intend to explore this area a lot more in the future, including looking at what ideas we can borrow from previous data integration work.

5. RELATED WORK

Since mashups are a hot topic right now, there has been a lot of previous work done in this field.

5.1 Data Integration

The database community has done a huge amount of research on data integration – reliably connecting together data from different sources that might have very different schemas, and different ways of naming the same things. Recent interesting examples include SEMEX [3], DataSpaces [7], and Cohera [24]. We believe that many of the techniques developed by this work are applicable to Mash Maker. Indeed we see Mash Maker, and mashups in general, as being primarily about providing approachable environments that allow arbitrary unskilled users to easily apply existing data integration techniques to repurpose data from arbitrary existing data sources.

5.2 Mashup Creation as a Separate Activity

There are many mashup tools that allow one to create mashups by graphically combining data sources and operators together as graphical dataflow graphs or pipelines. Examples include Yahoo Pipes [19], Marmite [25], Microsoft Popfly [20], IBM QED Wiki [21], and Anthracite [1]. These are all very powerful tools, and there are interesting differences between them, however they all follow a broadly similar model in which a reasonably skilled user creates a mashup by visually connecting components together, with the intention of creating a new site that users can use. Mash Maker extends this work by integrating both the creation and the use of mashups with the normal browsing experience, and predicting what the user wants based on their past behavior. In addition, with the exception of Marmite, these are all server-based tools, which means they have to deal with the issues we discussed in Section 3.

There are also a number of mashup creation tools that work at a lower level. Tools like Google Mashup Editor [8], Plagger.org, Ning.com, Javascript Dataflow Architecture [14], and Web Mashup Scripting Language [22] give the user a lot of power over the

mashups that they create, at the expense of requiring the user to write some form of program.

5.3 Mashups as Browsing

Operator [18] and Miro [6] are browser extensions that suggest actions to be performed on items they have found on the current web page. Operator looks for data tagged with microformats [15] and Miro uses a sophisticated data detector. Miro allows users to teach it new operations using a "program by example" interface. Mash Maker goes beyond what is possible with these tools by adding content to pages, and allowing one to create complex composite mashups that go beyond applying a single operation to the elements on a page.

GreaseMonkey[1] allows users to write scripts that can arbitrarily change the behavior of websites. If a user visits a web page for which they have registered a script, the greasemonkey script will run and can do pretty much anything to the page, including adding information to the page from other sites, or bringing in extra information. Many greasemonkey scripts provide behavior that is equivalent to, or superior to that which is achievable with Mash Maker mashups. GreaseMonkey provides mashup-writers with enormous power, at the cost of requiring them to write their mashups as Javascript programs.

There are also a number of mashup tools that excel in creating a particular kind of mashup. Google MyMaps and Microsoft MapCruncher[2] make it easy for end users to create mashups involving maps and Swivel.com makes it very easy for end users to create graph mashups from multiple data tables.

5.4 Semantic Web Browsers

Like Mash Maker, semantic web browsers such as Tabulator [2] and PiggyBank [13] are implemented as FireFox extensions, and allow one to browse data that can be found by following links on a page. PiggyBank allows one to create extractors using Solvent, which is similar to our extractor editor. Mash Maker improves on these tools by allowing one to add new information directly to the current page, rather than having to use a new interface. Mash Maker also extends this previous work by allowing users to compute new data from the data they have available, rather than being restricted to finding information by following links.

There are many other semantic web browsers. For example Haystack [9] is implemented as a stand-alone application and OntoWiki [10], DISCO [4], mSpace [16], and OpenLink [17] are implemented as web applications. All semantic web browsers we are aware of treat semantic web browsing as a separate task, rather than augmenting the normal browsing interface with semantic data.

5.5 Widgets

A huge number of sites exist that provide widgets that can be embedded into pages to show visualization of data. Google Widgets, ClearSpring.com, Widsets.com, WidgetBox.com, and Apple's Dashboard allow users to write small graphical web widgets and then lay them out together on a screen. DataMashups.com additionally allows users to connect these widgets together.

The difference between Mash Maker widgets and these other widget systems is more in their intended purpose, rather than the underlying architecture. Mash Maker intends that widgets be used to visualize potentially arbitrary RDF data, rather than being loosely parameterised representations of a particular web site.

[1] `http://diveintogreasemonkey.org`

[2] `http://research.microsoft.com/mapcruncher`

5.6 Content Suggesters

Many tools exist that try to understand the user's interests and suggest things that they might want. StumbleUpon.com is a browser toolbar that suggests web sites that one might be interested in. Last.fm and Pandora.com are internet radio stations that try to play songs that the user would be interested in. Amazon.com has a product suggestion system that suggests products that you might be interested in, based on past behavior.

6. CONCLUSIONS

Mash Maker as it is now is just a small step toward our eventual vision. Our intention is to move toward a personal proactive internet in which a user's computing devices anticipate what the users wants and make use of semantic information on the internet to present them with the information they want, presented the way they want it, while requiring a minimum of interaction.

Mash Maker is currently available as a limited "technology preview release". Although use is currently invite only, members of the public can sign up online to be put on a waiting list to be sent an invite when we want more testers. Visit the URL below to try out Intel®Mash Maker:

`http://mashmaker.intel.com`

Acknowledgments

This work has benefited from the input of many people. Particular thanks should go to Kulki Dattatraya, David Gay, Badari Kommandur, Eric Paulos, Rusty Sears, Ian Smith, and K Sridharan.

7. REFERENCES

[1] Anthracite. `http://www.metafly.com/products/anthracite`.

[2] T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets. Tabulator: Exploring and analysing linked data on the semantic web. In *Proceedings of the 3rd International Semantic Web User Interaction Workshop*, 2006.

[3] Y. Cai, X. L. Dong, A. Halevy, J. M. Liu, and J. Madhavan. Personal information management with SEMEX. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 921–923, New York, NY, USA, 2005. ACM Press.

[4] Disco - hyperdata browser. `http://sites.wiwiss.fu-berlin.de/suhl/bizer/ng4j/disco/`.

[5] R. Ennals and D. Gay. User-friendly functional programming for web mashups. In *ICFP '07: Proceedings of the 2007 ACM SIGPLAN international conference on Functional programming*, pages 223–234, New York, NY, USA, 2007. ACM Press.

[6] A. Faaborg and H. Lieberman. A goal-oriented web browser. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 751–760, New York, NY, USA, 2006. ACM Press.

[7] M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: A new abstraction for information management. In *SIGMOD Record*, 2005.

[8] Google mashup editor. `http://editor.googlemashups.com`.

[9] Haystack project. `http://groups.csail.mit.edu/haystack/`.

[10] M. Hepp, D. Bachlechner, and K. Siorpaes. Ontowiki: community-driven ontology engineering and ontology usage

based on wikis. In *WikiSym '06: Proceedings of the 2006 international symposium on Wikis*, 2006.

[11] E. V. Hippel. *Democratizing Innovation*. MIT Press, 2006.

[12] HTML 4.01 specification. `http://www.w3.org/TR/REC-html40/`.

[13] D. Huynh, S. Mazzocchi, and D. Karger. Piggy bank: Experience the semantic web inside your browser. In *Proceedings of the 4th International Semantic Web Conference*, 2005.

[14] S. C. S. Lim and P. Lucas. Jda: a step towards large-scale reuse on the web. In *OOPSLA '06: Companion to the 21st ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, pages 586–601, New York, NY, USA, 2006. ACM Press.

[15] Microformats. `http://microformats.org`.

[16] mspace. `http://mspace.fm`.

[17] OpenLink RDF Browser. `http://demo.openlinksw.com/DAV/JS/rdfbrowser/index.html`.

[18] Introducing operator. `http://labs.mozilla.com/2006/12/introducing-operator`.

[19] Yahoo Pipes. `http://pipes.yahoo.com`.

[20] Microsoft popfly. `http://popfly.com`.

[21] Qedwiki. `http://services.alphaworks.ibm.com/qedwiki/`.

[22] M. Sabbouh, J. Higginson, D. Gagne, and S. Semy. Web mashup scripting language (poster). In *16th International World Wide Web Conference*, 2007.

[23] SPARQL Query Language for RDF. `http://www.w3.org/TR/rdf-sparql-query/`.

[24] M. Stonebraker and J. M. Hellerstein. Content integration for e-business. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, 2001.

[25] J. Wong and J. Hong. Marmite: end-user programming for the web. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 1541–1546, New York, NY, USA, 2006. ACM Press.